

# Protein Hypernetworks: a Logic Framework for Interaction Dependencies and Perturbation Effects in Protein Networks

Johannes Köster<sup>\*†</sup>

Eli Zamir<sup>†‡</sup>

Sven Rahmann<sup>\*‡</sup>

Inofficial Preprint, June 15, 2011

**Motivation:** Protein interactions are fundamental building blocks of biochemical reaction systems underlying cellular functions. The complexity and functionality of such systems emerge not from the protein interactions themselves but from the dependencies between these interactions. Therefore, a comprehensive approach for integrating and using information about such dependencies is required.

**Results:** We present an approach for endowing protein networks with interaction dependencies using propositional logic, thereby obtaining *protein hypernetworks*. First we demonstrate how this framework straightforwardly improves the prediction of protein complexes. Next we show that modeling protein perturbations in hypernetworks, rather than in networks, allows to better infer the functional necessity of proteins for yeast. Furthermore, hypernetworks improve the prediction of synthetic lethal interactions in yeast, indicating their capability to capture high-order functional relations between proteins.

**Conclusion:** Protein hypernetworks are a consistent formal framework for modeling dependencies between protein interactions within protein networks. First applications of protein hypernetworks on the yeast interactome indicate their value for inferring functional features of complex biochemical systems.

**Availability:** Data and software is publicly available at <http://www.rahmannlab.de/research/hypernetworks>.

**Contact:** Eli.Zamir@mpi-dortmund.mpg.de,  
Sven.Rahmann@tu-dortmund.de

## 1 Introduction

A fundamental challenge in systems biology is understanding how cellular functions emerge from the collective action of interacting proteins. Ultimately such understanding could be achieved through a complete quantitative biochemical description of the system, including the concentrations and spatial distribution of all involved proteins and the kinetic constants of their interactions (Hughey *et al.*, 2010; Kholodenko, 2006). However, despite the progress in technologies for measuring these parameters in cells, completing such a description for large intracellular biochemical systems is still beyond reach. In a complementary front, high-throughput protein-protein interaction (PPI) detection techniques, including yeast two-hybrid and mass spectrometry (Walther and Mann, 2010; Parrish *et al.*, 2006), can provide static snapshots of complete interactomes, as demonstrated with several model organisms. The obtained information is typically modeled as networks – simple graphs with nodes and edges corresponding to the proteins and their interactions, respectively. However, such a data structure cannot represent information about how protein interactions depend on each other.

A key mechanism generating interaction dependencies is allosteric regulation, in which a protein undergoes conformational change upon one interaction which affects its other interactions (Laskowski *et al.*, 2009). Another common type of interaction dependencies is mutual exclusiveness arising from steric hindrance that prevents

proteins from binding simultaneously to too close or identical protein domains. Protein interaction dependencies determine the properties of biochemical systems, and therefore it is essential to comprehensively consider them. Importantly, vast information about interaction dependencies can be already obtained through database mining, and can be further expanded by high-throughput experimental approaches (see Discussion). However, a comprehensive approach to integrate this knowledge for getting a better understanding of large biochemical systems is still required.

Recent studies indicate that considering mutual exclusiveness between interactions improves the quality of protein complex prediction in yeast (Ozawa *et al.*, 2010; Jung *et al.*, 2010). Here, we further expand and generalize this potential by enabling on one hand the integration of diverse types of interaction dependencies and on the other hand the exploration of different aspects of the system. We use *propositional logic* to model interaction constraints, and provide a flexible framework for their system-wise representation, called *protein hypernetworks* (Section 2). Next, we show how to mine hypernetworks for useful information, exemplified here as improving the quality of protein complex prediction (Section 3). Furthermore, our approach allows ranking the importance of each protein in a biochemical system based not only on its interactions but also on their dependencies. We demonstrate that such considerations help predicting which proteins are essential for yeast viability (Section 4). Finally, we discuss how our approach synergizes with current efforts to obtain system-level understanding of complex biochemical systems.

## 2 Modeling Approach

### 2.1 Protein Hypernetworks

A protein network is commonly described as an undirected graph  $(P, I)$  with a vertex  $p \in P$  for each protein and an undirected edge  $\{p, p'\} \in I$  for each possible interaction. We first develop an approach for incorporating interaction dependencies into this description, using propositional logic formulas.

The propositional logic  $\mathfrak{Prop}(Q)$  is the set of all propositional logic formulas over the propositions  $Q$  (the atomic units of the logic). This is the smallest set of formulas such that  $q$  itself is a formula for all  $q \in Q$  and that is closed under the following operations: For  $\phi, \phi' \in \mathfrak{Prop}(Q)$ , all of  $\neg\phi$ ,  $\phi \wedge \phi'$ ,  $\phi \vee \phi'$ , and  $\phi \Rightarrow \phi'$  are in  $\mathfrak{Prop}(Q)$  as well. The operators  $\neg, \wedge, \vee, \Rightarrow$  have the usual semantics “not”, “and”, “or”, and “implies”, respectively. Note that the implication  $\phi \Rightarrow \phi'$  is equivalent to  $(\neg\phi \vee \phi')$ . As propositions  $Q$ , we use both proteins  $P$  and interactions  $I$ , so  $Q := P \cup I$ . A constraint is a formula with a particular structure over these propositions.

**Definition 1 (Constraint).** A constraint is a propositional logic formula of the form  $q \Rightarrow \psi$  with  $q \in P \cup I$  and  $\psi \in \mathfrak{Prop}(P \cup I)$ . With  $\mathfrak{C}(P \cup I) \subseteq \mathfrak{Prop}(P \cup I)$  we denote the set of all constraints.

A constraint  $q \Rightarrow \psi$  restricts the satisfiability of  $q$  by the satisfiability of  $\psi$ . In other words: if  $q$  is satisfied, then the same has to hold for  $\psi$ . A constraint  $q \Rightarrow \psi$  is equivalent to the disjunction  $\neg q \vee \psi$ . We call the disjunct  $\neg q$  the *default* or *inactive* case for the obvious reason that if  $q$  is not true, then  $\psi$  does not need to be satisfied. For example (see Fig. 1a), the dependency of an interaction  $i$  on an allosteric effect due to a scaffold interaction  $j$  can be formulated by the constraint  $i \Rightarrow j$ . Mutual exclusiveness of two interactions  $i, j \in I$  can be modelled by the two constraints  $i \Rightarrow \neg j$  and  $j \Rightarrow \neg i$ . The usage of propositional logic allows also to define constraints of higher order: An interaction  $i$  could be either

<sup>\*</sup>Bioinformatics for High-Throughput Technologies, Algorithm Engineering, Computer Science 11, TU Dortmund, Germany

<sup>†</sup>Max Planck Institute of Molecular Physiology, Dortmund, Germany

<sup>‡</sup>to whom correspondence should be addressed

dependent on two scaffold interactions  $j_1$  and  $j_2$  or compete with an interaction  $j_3$ , modeled by the constraint  $i \Rightarrow ((j_1 \wedge j_2) \vee \neg j_3)$ .

Now, we can define protein hypernetworks as a set of proteins (nodes) connected by interactions (edges) extended by a set of constraints (dependencies between nodes or edges):

**Definition 2** (Protein Hypernetwork). *Let  $P$  and  $I$  be sets of proteins and interactions. Let  $C \subseteq \mathcal{C}(P \cup I)$  be a set of constraints that contains the default constraints  $i \Rightarrow p \wedge p'$  for each interaction  $i = \{p, p'\} \in I$ . Then the triple  $(P, I, C)$  is called a protein hypernetwork.*

Fig. 1a shows an example protein hypernetwork. While a protein hypernetwork is not a hypergraph (with hyperedges) in the classical sense, the name is appropriate because the constraints describe dependencies between the edges, which could be explicitly transformed into hyperedges, e.g., using minimal network states (cf. Sec. 2.2).

## 2.2 Minimal Network States

Following the incorporation of constraints in a protein hypernetwork, we now explain how to sum and propagate their effects in the system. The key idea is that it is sufficient to examine the implications for each protein or interaction  $q \in P \cup I$  separately first, and then combine the information in a systematic way. We formalize this idea by defining sets of *minimal network states*. A minimal network state of  $q$  tells us which other proteins or interactions are *necessary* or *impossible* to occur simultaneously with  $q$ . For each  $q \in P \cup I$ , we define a *minimal network state formula*, for which we then find certain satisfying models, which in turn define minimal network states.

**Definition 3** (Minimal network state formula). *Let  $(P, I, C)$  be a protein hypernetwork. For  $q \in P \cup I$ , the minimal network state formula of  $q$  is*

$$MNS_{(P, I, C)}(q) := MNS(q) := q \wedge \bigwedge_{c \in C} c.$$

A solution for a propositional logic formula is captured by a *satisfying model* or *interpretation* given by a map  $\alpha : P \cup I \rightarrow \{0, 1\}$  that assigns a truth value to each proposition. A formula is *satisfiable* if any satisfying model exists. We assume that  $MNS(q)$  is satisfiable for all  $q \in P \cup I$ , i.e., each single protein or interaction by itself is compatible with all constraints.

For example, consider propositions  $Q = \{q_1, q_2\}$  and a formula  $\phi = \neg q_1 \wedge (q_1 \vee q_2)$ . The only satisfying model is  $\alpha : q_1 \mapsto 0, q_2 \mapsto 1$ . In the protein hypernetworks framework, we interpret a model  $\alpha$  as follows: A protein or interaction  $q$  is said to be *possible* in  $\alpha$  iff  $\alpha(q) = 1$ . All possible proteins and interactions may (but need not) exist simultaneously (spatially and temporally) in the cell.

There can be many satisfying models for  $MNS(q)$ . Among these, we wish to enumerate all *minimally constrained satisfying models* (MCSMs). A suitable method for finding them is the tableau calculus for propositional logic (Smullyan, 1995). In a nutshell, the tableau algorithm decomposes a formula into its parts. It accumulates conjuncts, branches on disjuncts, and backtracks when a contradiction is encountered. More details are given in Sec. S1 of the Supplement. For finding MCSMs, our custom implementation ensures that for each constraint  $q \Rightarrow \psi$  (i.e., disjunction  $\neg q \vee \psi$ ), the default case  $\neg q$  is explored first, and that  $\psi$  is expanded only if the constraint is necessarily active, in order to avoid artificially constrained models.

The general problem of deciding whether any given propositional logic formula  $\phi$  is satisfiable is NP-complete. However,  $MNS(q)$  has a special structure: it is a conjunction of a proposition and of (many) constraints. If all constraints are of a particularly simple structure, we can prove a linear running time; see Sec. S1 in the Supplement.

Each MCSM  $\alpha$  defines a minimal network state, consisting of both necessary and impossible entities. The intuition is that the necessary entities  $k$  are simply the “true” ones ( $\alpha(k) = 1$ ), and that the impossible entities are those that are *explicitly* forbidden by an active constraint.

**Definition 4** (Minimal Network State). *Let  $(P, I, C)$  be a protein hypernetwork and  $q \in P \cup I$ . Let  $\alpha$  be a MCSM of  $MNS(q)$ . We*

*define sets of necessary and impossible proteins or interactions, respectively, as*

$$\begin{aligned} Nec_\alpha &:= \{k \in P \cup I \mid \alpha(k) = 1\}, \\ Imp_\alpha &:= \{k \in P \cup I \mid \exists \text{ constraint } (q' \Rightarrow \psi) \in C \\ &\quad \text{with } \alpha(q') = 1 \text{ and } \psi \wedge k \text{ is unsatisfiable.}\}. \end{aligned}$$

*The pair  $(Nec_\alpha, Imp_\alpha)$  is called a minimal network state for  $q$  (belonging to the MCSM  $\alpha$ ).*

For each proposition  $q$ , there can be several minimal network states. We write  $M_q$  for the set of all minimal network states for  $q$ . We call  $M := M_{(P, I, C)} := \bigcup_q M_q$  the set of all minimal network states for all proteins and interactions.

Now we define a relation *clashing*, describing that two minimal network states cannot be combined without producing a conflict.

**Definition 5** (Clashing Minimal Network States). *Two minimal network states  $(Nec, Imp)$  and  $(Nec', Imp')$  are clashing iff  $Nec \cap Imp' \neq \emptyset$  or  $Imp \cap Nec' \neq \emptyset$ .*

As we prove in Theorem 1, in order to know if two proteins or interactions are simultaneously possible, it is sufficient to determine whether any pair of non-clashing minimal network states exists for them.

**Theorem 1.** *Let  $(P, I, C)$  be a protein hypernetwork. Let  $q, q' \in P \cup I$  be two proteins or interactions,  $q \neq q'$ . Assume that there exists a non-clashing pair of minimal network states  $(m, m') \in M_q \times M_{q'}$ . Then  $q$  and  $q'$  are possible simultaneously, i.e., the following formula is satisfiable.*

$$\xi := \left( \bigwedge_{c \in C} c \right) \wedge q \wedge q'.$$

*Proof.* Let  $m = (Nec, Imp) \in M_q$  and  $m' = (Nec', Imp') \in M_{q'}$  be non-clashing; we show that  $\xi$  is satisfiable by defining a satisfying model  $\alpha$ . Define  $True := Nec \cup Nec'$  and  $False := Imp \cup Imp'$ . Since  $m$  and  $m'$  are not clashing,  $True \cap False = \emptyset$ . Let  $\alpha(r) := 1$  for  $r \in True$ , and  $\alpha(r) := 0$  otherwise. We show that  $\alpha$  satisfies all parts of  $\xi$ .

The propositions  $q$  and  $q'$  in  $\xi$  are satisfied since  $q \in Nec$  and  $q' \in Nec'$ , so  $\alpha(q) = \alpha(q') = 1$ .

For each  $c^* = (r \Rightarrow \psi)$  in the conjunction  $\bigwedge_{c \in C} c$ , there may appear two cases:  $\alpha(r) = 0$  or  $\alpha(r) = 1$ . If  $\alpha(r) = 0$ , then  $c$  is satisfied regardless of the satisfaction of  $\psi$  because of the implication semantics. If  $\alpha(r) = 1$ , or equivalently  $r \in True$ , then  $r \in Nec$  or  $r \in Nec'$  (or both). First, consider the case that  $r \in Nec$ . By assumption,  $c^* \in C$  is then satisfied in  $\bigwedge_{c \in C} c \wedge q$ . Additionally, it is not clashing with  $q'$  because  $True \cap False = \emptyset$ . Therefore, it is also satisfied in  $\xi$ . The case  $r \in Nec'$  is analogous.  $\square$

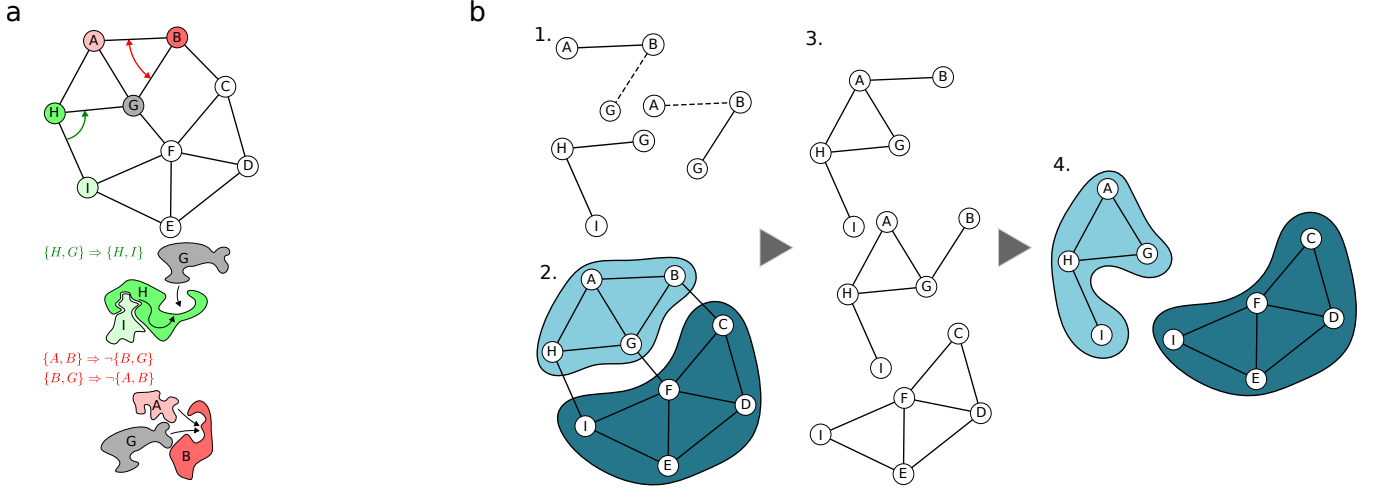
Minimal network states are the basis for further inferences on hypernetworks, as we demonstrate in Sections 3 and 4. First, however, we show that perturbations can be easily incorporated into the framework.

## 2.3 Inclusion of Perturbation Effects

The protein hypernetwork framework allows to systematically compute consequences of perturbations. We distinguish between *perturbed* and *affected* proteins or interactions: A perturbed one is the direct target of an experimental intervention which causes its complete removal from the system (e.g. by gene knock-down for proteins or point mutations for interactions), whereas an affected one is altered due to the propagation of the perturbation in the hypernetwork. Assume that proteins  $P_\downarrow \subseteq P$  and interactions  $I_\downarrow \subseteq I$  are perturbed, and thus removed from the system. The problem at hand is to compute all affected proteins and interactions. This is done by recursively removing minimal network states  $m = (Nec, Imp)$  that necessitate a perturbed or affected entity (protein or interaction)  $q \in Nec$ , while counting a protein or interaction as affected once it has no minimal network state left. Formally, we proceed as follows.

**Definition 6.** *Let  $M_q$  be the set of all minimal network states for entity  $q$  (Definition 4), and let  $M \subseteq M_{(P, I, C)}$  be any subset of all minimal network states. For a set of entities  $A \subseteq P \cup I$ , let*

$$\bar{M}_A := \{(Nec, Imp) \in M \mid A \cap Nec \neq \emptyset\}$$



**Figure 1:** (a) Principle of protein-hypernetwork construction. A protein network (nodes and black edges) is overlaid with two interaction constraints: mutual exclusive interactions (top arc arrow) and activating allosteric interactions (lower arc arrow). Propositional logic formulas for these interaction constraints and the molecular mechanism generating them are shown. The protein hypernetwork is the plain network together with all such propositional logic constraints. (b) Protein complex prediction in four steps (see Sec. 3.2 for algorithmic details): (1) Computation of minimal network states (Sec. 2.2), (2) prediction of initial protein complexes, (3) computation of simultaneously possible protein subnetworks, (4) refinement of the predicted complexes.

be the set of minimal network states from  $M$  that become invalid when any entity in  $A$  is perturbed. Let  $R(A, M) := M \setminus \bar{M}_A$  be the remaining set of minimal network states. Let  $Q(A, M) := \{q \in P \cup I \mid M_q \cap R(A, M) = \emptyset\}$  be the set of entities for which no minimal network state is left.

We recursively define a map  $\rho$  that maps a set of perturbed entities and a set of minimal network states to the set of affected entities. Let  $\rho : 2^{P \cup I} \times 2^{M(P, I, C)} \rightarrow 2^{P \cup I}$  be defined by

$$\rho(A, M) := \begin{cases} \emptyset & \text{if } A = \emptyset, \\ A \cup \rho(Q(A, M), R(A, M)) & \text{otherwise.} \end{cases}$$

Let  $(P, I, C)$  be a protein hypernetwork with perturbations  $P_\downarrow \subseteq P$  and  $I_\downarrow \subseteq I$  and minimal network states  $M_{(P, I, C)}$ . Then

$$Q_\downarrow := \rho(P_\downarrow \cup I_\downarrow, M_{(P, I, C)})$$

is the set all affected proteins and interactions.

This provides a module that enables any algorithm that makes predictions based on protein networks to be applied also on a perturbed network, considering the dependencies between interactions.

### 3 Result I: Hypernetworks Improve Prediction of Protein Complexes

#### 3.1 Rationale

When considering only the interactions between proteins, but not their dependencies, the prediction of protein complexes often relies on identifying dense regions in protein networks (Spirin and Mirny, 2003; Bader and Hogue, 2003; Li *et al.*, 2005). Indeed, algorithms for the prediction of complexes based on plain protein networks  $(P, I)$  like MCODE (Bader and Hogue, 2003) and LCMA (Li *et al.*, 2005) have been shown to provide reasonable results by detecting such dense regions. However, many of the complexes predicted in this way are false positives, since interaction dependencies do not allow their assembly. Along this line, it was recently shown that consideration of mutual exclusiveness between interactions improve the quality of protein complex prediction (Jung *et al.*, 2010).

Here, we first provide a general framework that can build on an arbitrary network-based complex prediction method and ensures that the predicted complexes do not violate arbitrary given interaction constraints within the hypernetwork. Thus, the framework is much more general than the work by Jung *et al.* (2010); in practice, however, the main problem is obtaining sufficiently many constraints (see Discussion). We demonstrate that our framework improves complex prediction on the yeast network in conjunction with the established constraints using the LCM algorithm (Li *et al.*, 2005) as an example network-based complex prediction method.

#### 3.2 Algorithm

The prediction of protein complexes in hypernetworks consists of four steps, illustrated in Fig. 1b. First, for each protein and interaction  $q \in P \cup I$ , the set of minimal network states  $M_q$  is obtained. Then, with a network based complex prediction algorithm, an initial set of protein complexes is predicted. Each complex  $c$  is given as a subnetwork  $(P_c, I_c)$ .

The third step is more complicated. Let  $M_c := \bigcup_{q \in P_c \cup I_c} M_q$  be the set of minimal network states of the complex's entities. We want to combine the individual states without introducing clashes, as formalized by the following definition.

**Definition 7** (Maximal combination of minimal network states). For a complex  $c$ , a set  $M \subseteq M_c$  is called a maximal combination of minimal network states iff (1) there exists no clashing pair of minimal network states in  $M$ , and (2) the inclusion of any further minimal network state from  $M_c$  would result in a clashing pair.

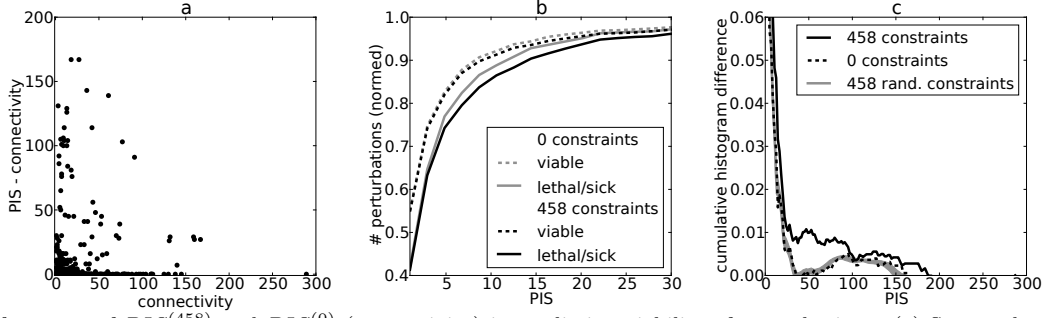
All maximal combinations of minimal network states for a given complex  $c$  can be obtained by recursively building a tree of minimal network states to be removed from  $M_c$ . The root of the tree is annotated with  $M_c$ ; each other node is annotated with a remaining set  $M$ . If  $M$  does not contain any pair of clashing states, the node is a leaf, and  $M$  is added to the result set of maximal combinations. Otherwise, we take any  $m$  with clashing  $m', m'', \dots$  and branch off two children which remove  $m$  on the one hand, and remove  $m', m'', \dots$  on the other hand. The tree is explored in a depth-first manner, checking for redundancies in each node. Let  $\mathcal{M}_c \subseteq 2^{M_c}$  be the set of all found maximal combinations of minimal network states. Its cardinality equals the number of non-redundant leaves in the removal tree. For each maximal combination  $M \in \mathcal{M}_c$ , we generate the corresponding subnetwork of  $(P, I)$ .

**Definition 8** (Simultaneous Protein Subnetwork). Let  $M \in \mathcal{M}_c$  be a maximal combination of minimal network states for complex  $c$ . Let  $P_M$  be the set of all necessary proteins and  $I_M$  the set of all necessary interactions in  $M$ , i.e.,  $P_M := P \cap \bigcup_{(Nec, Imp) \in M} Nec$  and  $I_M := I \cap \bigcup_{(Nec, Imp) \in M} Nec$ . Then  $(P_M, I_M)$  is called a simultaneous protein subnetwork.

All proteins and interactions in  $(P_M, I_M)$  may exist simultaneously in the context of the protein hypernetwork  $(P, I, C)$  because the minimal network states in  $M$  do not clash with each other. In comparison to the subnetwork for the network based predicted complex  $(P_c, I_c)$ , each subnetwork  $(P_M, I_M)$  may have lost and gained several interactions or proteins.

Finally, in the fourth step, we perform a network based complex prediction on each simultaneous protein subnetwork  $(P_M, I_M)$





**Figure 3:** Performance of  $PIS^{(458)}$  and  $PIS^{(0)}$  (connectivity) in predicting viability of perturbations. (a) Scatterplot of connectivity against difference PIS minus connectivity; note that always  $PIS \geq$  connectivity. (b) Cumulative distribution function (cdf) of  $PIS^{(458)}$  and  $PIS^{(0)}$  for viable and lethal/sick perturbations (axes span region of distinguishable values). (c) Differences between the cdfs (for details see Supplement, Sec. S4.2) of viable and lethal/sick perturbations, for 0 constraints, 458 constraints, and 458 random constraints. For the latter, 100 samples of 458 random constraints were drawn (see Supplement Sec. S3), and the figure shows the area between the mean plus minus one standard deviation. Note that the  $PIS^{(458)}$  curve is always above the  $PIS^{(0)}$  curve, and that the  $PIS^{(0)}$  curve agrees well with the randomized curve, indicating the contribution of constraints to a better discrimination between viable and lethal/sick perturbations.

$Q_{\downarrow}$  in the minimal network state graph. Define a distance function  $dist_{Q_{\downarrow}} : R_{\downarrow} \rightarrow \mathbb{N}$  such that  $dist_{Q_{\downarrow}}(q)$  is the shortest path length  $G_{MNS}$  between  $q$  and any node in  $Q_{\downarrow}$  (this is well-defined because consideration is restricted to reachable nodes  $q$ ). The perturbation impact score of the set  $Q_{\downarrow}$  is defined as

$$PIS_{(P,I,C)}(Q_{\downarrow}) := \sum_{q \in R_{\downarrow}} dist_{Q_{\downarrow}}(q).$$

The PIS models the idea that a protein or interaction is likely to be more important the further its perturbation propagates through the network. Of course,  $dist_{Q_{\downarrow}}(q)$  can be computed by a standard breadth-first search (BFS) on  $G_{MNS}$  beginning with the nodes in  $Q_{\downarrow}$ .

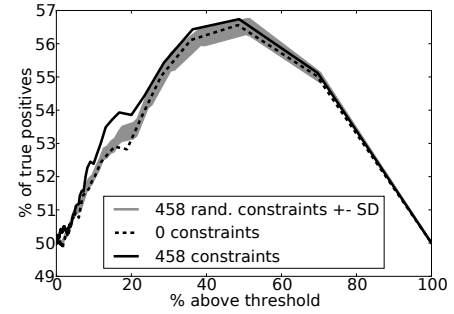
When computing the perturbation impact score  $PIS_{(P,I,C)}(\{p\})$  of a single protein  $p$  that does neither appear in a constraint itself nor has a neighbor that does, its score is equal to its connectivity. Since connectivity was shown to correlate with the functional importance of proteins (Jeong *et al.*, 2001), the incorporation of constraints in the PIS can be expected to further enhance this prediction quality. As will be shown, PIS allows also to measure the impact of combination of perturbations, thereby to infer functional relations between proteins such as synthetic lethality.

## 4.2 Experiments

We evaluated PIS against plain connectivity as estimators for the functional importance of proteins. Recalling that PIS without constraints equals connectivity, we let  $PIS^{(0)} := PIS_{(P,I,\emptyset)}$  be the connectivity, and  $PIS^{(458)} := PIS_{(P,I,C)}$  the score for the constrained CYGD-based yeast protein hypernetwork  $(P,I,C)$  as in Sec. 3.3. By definition,  $PIS^{(458)}$  is always equal to or higher than connectivity (Fig. 3a).

We assume that perturbation of functionally important proteins is more likely to produce sickness or cell death. Accordingly, for benchmarking, we classified perturbations as *lethal/sick* and *viable* according to the Saccharomyces Genome Database of null mutant phenotypes (1-4-2011, Cherry *et al.* (1998); see Supplement, Sec. S4.1, for details). From the distribution of PIS for both of these classes of perturbations (Fig. 3b) it is apparent that proteins resulting lethal/sick null mutants tend to have a higher PIS, regardless of the consideration of constraints.

Therefore, we investigated more closely the increase in PIS caused by the interaction constraints. While 7% of the lethal or sick perturbations exhibit an increased PIS upon consideration of constraints, only 2% of the viable ones do. To measure the separation between the classes we subtract the cdf of lethal/sick from that for viable. The higher this difference, the better the separation (for details see Supplement, Sec. S4.2). Fig. 3c shows this measure for 0, 458 and as a mean for 100 samples of 458 random constraints. The application of 458 random constraints does not alter the difference compared to 0 constraints. In contrast, the application of the true 458 constraints increases the difference – most obvious for a PIS between 20 and

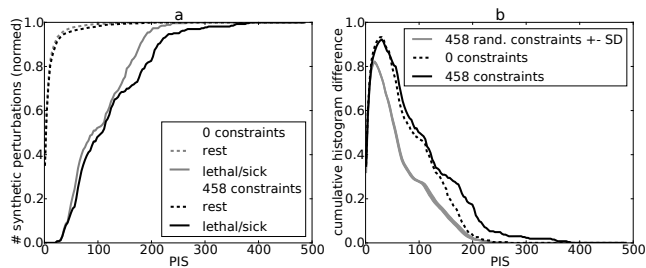


**Figure 4:** Quality of the prediction of lethal/sick and viable perturbations as a function of the threshold used for the classification, for  $PIS^{(0)}$ ,  $PIS^{(458)}$  and 1000 samples of 458 random constraints (see Supplement Sec. S3). x-axis: threshold given as percentage of perturbations above a certain PIS or connectivity value; y-axis: combined ratio of true positives for lethal/sick and viable perturbations  $(TP_{\text{lethal}}/P_{\text{lethal}} + TP_{\text{viable}}/P_{\text{viable}})/2$ .

100 – and hence improves the discrimination between lethal/sick and viable perturbations.

In general, PIS and plain connectivity agree for most proteins (overlaid points along the horizontal axis in Fig. 3a), because only a few constraints are available so far. However, for several proteins the difference is striking. For example the yeast protein SME1, which is required for mRNA splicing and whose perturbation is lethal to the cell (Guldener *et al.*, 2005), has only 6 binding partners in the CYGD – a relatively low connectivity that is not correlated with its biological importance. With the application of constraints, the PIS of SME1 increases to 111 and therefore correctly suggests that perturbation of this protein would be lethal. Counterexamples, where the introduction of constraints wrongly increases the PIS of a viable protein, also exist, however they are a minority, as indicated by the fact that constraints improve the overall performance of PIS (e.g. Fig. 3c).

To illustrate the use of PIS to predict the functional importance of a protein, we predicted lethal/sick and viable perturbations by systematically applying a threshold  $t$  to the PIS. If a protein had a PIS of at least  $t$  we predicted its perturbation to be lethal/sick, while we predicted it to be viable for a PIS less than  $t$ . Fig. 4 shows the prediction quality for different thresholds, when using 0 or 458 constraints or 1000 samples of 458 random constraints. To ensure the comparability of thresholds  $t$ , they are expressed as the percentage of proteins reaching or exceeding the PIS or connectivity such that half of the proteins reach or exceed it). It can be seen that every non-trivial threshold performs better than the trivial ones that predict no ( $t = 0$ ) or all ( $t = 100$ ) perturbations to be lethal/sick. Further, the application of the true constraints provides an improved prediction (especially below  $t = 20$ ) in comparison to



**Figure 5:** PIS is an indicator for lethal/sick synthetic perturbations. (a) cumulative distributions functions (cdfs) of PIS for synthetic perturbations with 0 and 458 constraints, distinguished between lethal/sick (as defined by Tong *et al.* (2004)) and the rest of synthetic perturbations. (b) Differences of cdfs (cf. Supplement, Sec. S4.2) of viable and lethal/sick perturbations, for 0 and 458 constraints and 100 samples of 458 random constraints (cf. Supplement, Sec. S3). Note that for  $PIS > 50$  the  $PIS^{(458)}$  curve lies above the  $PIS^{(0)}$  curve, indicating the contribution of constraints to a better discrimination between viable and lethal/sick synthetic perturbations.

random constraints (Fig. 4).

Many null mutations do not affect viability when occurring alone, but become lethal when occurring together with another specific null mutation (i.e. synthetic lethality), indicating functional buffering and relation between the corresponding proteins (Tong *et al.*, 2001, 2004). To evaluate if PIS can capture these pair-wise protein relations, we investigated its ability to predict synthetic perturbations by calculating  $PIS_{(P,I,C)}(\{p, p'\})$  for every pair of proteins  $p, p' \in P$ . Note that without constraints, PIS here equals counting the union of neighbors of two proteins. Fig. 5 shows that PIS provides a striking separation between lethal/sick and viable perturbations (as defined by Tong *et al.* (2004)). Further, the application of constraints induces a shift of the scores toward higher values, that results in an improved discrimination between viable and lethal/sick synthetic perturbations (Fig. 5b). In contrast, using 100 samples of 458 random constraints again does not provide an improvement, and even decreases the discrimination quality.

We conclude that PIS is an improvement over connectivity as a predictive measure for functional importance, as it allows to integrate interaction constraints from hypernetworks. Since only 2.7% of interactions are constrained in our experiments, improvements by constraints are naturally small here. We expect the capability of PIS to discriminate between lethal/sick and viable perturbations to further increase as information about additional interaction constraints will become available.

## 5 Discussion

The dependencies of protein interactions encode the capability of PPI systems to process information and execute cellular decisions. We developed an approach to unfold this dimension of information by incorporating interaction constraints generated by allosteric regulations and competitive binding. On the level of individual proteins, competition between interactions on the same binding domain leads to their complete mutual exclusiveness. Similarly, allosteric regulations typically generate all-or-none switches between a non-binding and a binding state (Laskowski *et al.*, 2009). Therefore, propositional logic can capture perfectly these fundamental processes, and additionally facilitates their algorithmic integration. Similarly, protein hypernetworks can incorporate regulations of protein interactions by post-translational modifications (e.g. phosphorylation), as these are often on/off switches describable by propositional logic (e.g.  $\{A, B\} \Rightarrow \{B, PO_4\}$  states that B has to be phosphorylated to allow its interaction with A). The temporal expression and spatial distribution of intracellular proteins, which were shown to be valuable dimensions of information (Han *et al.*, 2004; Walther and Mann, 2010), can also be incorporated into the hypernetwork framework by discretizing them in time (e.g. cell-cycle phases or developmental stages) and space (e.g. by sub-cellular compartment or by tissue).

The question addressed in this work is how to use information

about interaction dependencies, rather than how to collect it. Nevertheless, it should be noted that a significant amount of information about interaction dependencies can already be obtained through curation from literature. Along this line, we are currently developing a text-mining tool to assist the identification of publications reporting interaction dependencies. As for future publications, since automatic curation of protein interactions is facilitated by a structured text format (Leitner *et al.*, 2010; Ceol *et al.*, 2008a), our work motivates its usage to report interaction dependencies (see Supplementary Sec. S5.1). Mutual exclusiveness between protein interactions can also be inferred from protein-domain-annotated interactome databases (Ooi *et al.*, 2010) or in-silico docking modeling (Wass *et al.*, 2011; Mosca *et al.*, 2009). Finally, high-throughput quantification of protein interactions at domain resolution and methods for monitoring high-order interactions (Jain *et al.*, 2011; Hruby *et al.*, 2011; Heinze *et al.*, 2004) would provide comprehensive identification of interaction dependencies.

Here, we illustrated that even constraining less than 3% of the interactions in the CYGD is sufficient to improve complex prediction, consistently with previous results (Jung *et al.*, 2010). It is expected that the actual fraction of constrained interactions is much higher, to allow a dynamic and functional yeast interactome. These additional interaction constraints would be due to allosteric regulations (Laskowski *et al.*, 2009), generation and elimination of binding sites upon protein phosphorylation and dephosphorylation (Seet *et al.*, 2006) and more cases of mutually exclusive interactions.

We proposed a perturbation impact score that provides a measure for a protein's importance within a hypernetwork. We have shown that this measure provides improvements to the prediction of functionally important proteins compared to the investigation of plain connectivity due to the usage of interaction dependencies as constraints. As more constraints get reported, the measure should help to rationally design perturbation experiments for network analysis (Zamir and Bastiaens, 2008) and provide mechanistic insights into large PPI systems.

Our data and software for protein hypernetworks are available; please refer to the Supplement (Sec. S5) for implementation details.

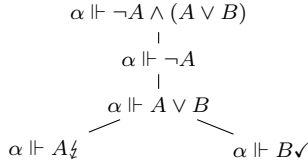
## References

- Altaf-Ul-Amin, M., Shinbo, Y., Mihara, K., Kurokawa, K., and Kanaya, S. (2006). Development and implementation of an algorithm for detection of protein complexes in large interaction networks. *BMC Bioinformatics*, **7**(1).
- Bader, G. and Hogue, C. (2003). An automated method for finding molecular complexes in large protein interaction networks. *BMC Bioinformatics*, **4**(1), 2–29.
- Brandes, U., Eiglsperger, M., Herman, I., Himsolt, M., and Marshall, M. (2002). GraphML progress report: Structural layer proposal. pages 501–512. Springer-Verlag.
- Bray, T., Paoli, J., and Sperberg-McQueen, C. M. (1998). Extensible markup language (XML) 1.0. Available via the World Wide Web at <http://www.w3.org/TR/1998/REC-xml-19980210>.
- Carlisle, D., Ion, P., Miner, R., and Poppelier, N. (2003). Mathematical Markup Language (MathML) 2.0. Available via the World Wide Web at <http://www.w3.org/TR/MathML2/>.
- Ceol, A., Chatr-Aryamontri, A., Licata, L., and Cesareni, G. (2008a). Linking entries in protein interaction database to structured text: the FEBS letters experiment. *FEBS Lett*, **582**(8), 1171–7.
- Ceol, A., Chatr-Aryamontri, A., Licata, L., and Cesareni, G. (2008b). Linking entries in protein interaction database to structured text: the FEBS Letters experiment. *FEBS letters*, **582**(8), 1171–1177.
- Cherry, J. M., Adler, C., Ball, C., Chervitz, S. A., Dwight, S. S., Hester, E. T., Jia, Y., Juvik, G., Roe, T., Schroeder, M., Weng, S., and Botstein, D. (1998). SGD: Saccharomyces Genome Database. *Nucleic acids research*, **26**(1), 73–79.
- Cote, R., Jones, P., Apweiler, R., and Hermjakob, H. (2006). The Ontology Lookup Service, a lightweight cross-platform tool for controlled vocabulary queries. *BMC Bioinformatics*, **7**(1), 97+.



- Dowling, W. (1984). Linear-time algorithms for testing the satisfiability of propositional horn formulae. *The Journal of Logic Programming*, **1**(3), 267–284.
- Feng, J., Jiang, R., and Jiang, T. (2010). A Max-Flow Based Approach to the Identification of Protein Complexes Using Protein Interaction and Microarray Data. *IEEE/ACM transactions on computational biology and bioinformatics / IEEE, ACM*.
- Güldener, U., Münsterkötter, M., Kastenmüller, G., Strack, N., van Helden, J., Lemer, C., Richelles, J., Wodak, S. J., García-Martínez, J., Pérez-Ortín, J. E., Michael, H., Kaps, A., Talla, E., Dujon, B., André, B., Souciet, J. L., De Montigny, J., Bon, E., Gaillardin, C., and Mewes, H. W. (2005). CYGD: the comprehensive yeast genome database. *Nucleic Acids Research*, **33**(Supplement 1), D364–D368.
- Han, J.-D. J., Bertin, N., Hao, T., Goldberg, D. S., Berriz, G. F., Zhang, L. V., Dupuy, D., Walhout, A. J. M., Cusick, M. E., Roth, F. P., and Vidal, M. (2004). Evidence for dynamically organized modularity in the yeast protein-protein interaction network. *Nature*, **430**(6995), 88–93.
- Heinze, K. G., Jahnz, M., and Schwille, P. (2004). Triple-color coincidence analysis: one step further in following higher order molecular complex formation. *Biophys J*, **86**(1 Pt 1), 506–16.
- Hruby, A., Zapatka, M., Heucke, S., Rieger, L., Wu, Y., Nussbaumer, U., Timmermann, S., Dünkler, A., and Johnsson, N. (2011). A constraint network of interactions: protein-protein interaction analysis of the yeast type ii phosphatase *ptc1p* and its adaptor protein *nbp2p*. *J Cell Sci*, **124**(Pt 1), 35–46.
- Hucka, M., Finney, A., Sauro, H. M., Bolouri, H., Doyle, J. C., Kitano, H., , the rest of the SBML Forum: Arkin, A. P., Bornstein, B. J., Bray, D., Cornish-Bowden, A., Cuellar, A. A., Dronov, S., Gilles, E. D., Ginkel, M., Gor, V., Goryanin, I. I., Hedley, W. J., Hodgman, T. C., Hofmeyr, J. H., Hunter, P. J., Juty, N. S., Kasberger, J. L., Kremling, A., Kummer, U., Le Novère, N., Loew, L. M., Lucio, D., Mendes, P., Minch, E., Mjolsness, E. D., Nakayama, Y., Nelson, M. R., Nielsen, P. F., Sakurada, T., Schaff, J. C., Shapiro, B. E., Shimizu, T. S., Spence, H. D., Stelling, J., Takahashi, K., Tomita, M., Wagner, J., and Wang, J. (2003). The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics*, **19**(4), 524–531.
- Hughey, J. J., Lee, T. K., and Covert, M. W. (2010). Computational modeling of mammalian signaling networks. *Wiley Interdiscip Rev Syst Biol Med*, **2**(2), 194–209.
- Jain, A., Liu, R., Ramani, B., Arauz, E., Ishitsuka, Y., Ragunathan, K., Park, J., Chen, J., Xiang, Y. K., and Ha, T. (2011). Probing cellular protein complexes using single-molecule pull-down. *Nature*, **473**(7348), 484–8.
- Jeong, H., Mason, S. P., Barabasi, A. L., and Oltvai, Z. N. (2001). Lethality and centrality in protein networks. *Nature*, **411**(6833), 41–42.
- Jung, S. H., Hyun, B., Jang, W.-H., Hur, H.-Y., and Han, D.-S. (2010). Protein complex prediction based on simultaneous protein interaction network. *Bioinformatics*, **26**(3), 385–391.
- Kholodenko, B. N. (2006). Cell-signalling dynamics in time and space. *Nat Rev Mol Cell Biol*, **7**(3), 165–76.
- Köster, J. (2011). Hypernetwork Markup Language 1.0. Available via the World Wide Web at <http://www.rahmannlab.de/research/hypernetworks/hypernetworkml>.
- Laskowski, R. A., Gerick, F., and Thornton, J. M. (2009). The structural basis of allosteric regulation in proteins. *FEBS Lett*, **583**(11), 1692–8.
- Leitner, F., Chatr-aryamontri, A., Mardis, S. A., Ceol, A., Krallinger, M., Licata, L., Hirschman, L., Cesareni, G., and Valencia, A. (2010). The febs letters/bioreactive ii.5 experiment: making biological information accessible. *Nat Biotechnol*, **28**(9), 897–9.
- Li, X.-L. L., Tan, S.-H. H., Foo, C.-S. S., and Ng, S.-K. K. (2005). Interaction graph mining for protein complexes using local clique merging. *Genome informatics*, **16**(2), 260–269.
- Li, Z. (2008). *Efficient and Generic Reasoning for Modal Logics*. Ph.D. thesis, School of Computer Science, University of Manchester, UK.
- Mosca, R., Pons, C., Fernández-Recio, J., and Aloy, P. (2009). Pushing structural information into the yeast interactome by high-throughput protein docking experiments. *PLoS Comput Biol*, **5**(8), e1000490.
- Ooi, H. S., Schneider, G., Chan, Y.-L., Lim, T.-T., Eisenhaber, B., and Eisenhaber, F. (2010). Databases of protein-protein interactions and complexes. *Methods Mol Biol*, **609**, 145–59.
- Ozawa, Y., Saito, R., Fujimori, S., Kashima, H., Ishizaka, M., Yanagawa, H., Miyamoto-Sato, E., and Tomita, M. (2010). Protein complex prediction via verifying and reconstructing the topology of domain-domain interactions. *BMC Bioinformatics*, **11**, 350.
- Parrish, J. R., Gulyas, K. D., and Finley, Jr, R. L. (2006). Yeast two-hybrid contributions to interactome mapping. *Curr Opin Biotechnol*, **17**(4), 387–93.
- Seet, B. T., Dikic, I., Zhou, M.-M., and Pawson, T. (2006). Reading protein modifications with interaction domains. *Nat Rev Mol Cell Biol*, **7**(7), 473–83.
- Selman, B., Levesque, H. J., and Mitchell, D. (1992). A New Method for Solving Hard Satisfiability Problems. In P. Rosenbloom and P. Szolovits, editors, *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 440–446, Menlo Park, California. AAAI Press.
- Smullyan, R. M. (1995). *First-Order Logic*. Dover Publications.
- Spirin, V. and Mirny, L. A. (2003). Protein complexes and functional modules in molecular networks. *Proceedings of the National Academy of Sciences of the United States of America*, **100**(21), 12123–12128.
- Tong, A. H., Evangelista, M., Parsons, A. B., Xu, H., Bader, G. D., Page, N., Robinson, M., Raghibizadeh, S., Hogue, C. W. V., Bussey, H., Andrews, B., Tyers, M., and Boone, C. (2001). Systematic Genetic Analysis with Ordered Arrays of Yeast Deletion Mutants. *Science*, **294**(5550), 2364–2368.
- Tong, A. H. Y. H., Lesage, G., Bader, G. D., Ding, H., Xu, H., Xin, X., Young, J., Berriz, G. F., Brost, R. L., Chang, M., Chen, Y., Cheng, X., Chua, G., Friesen, H., Goldberg, D. S., Haynes, J., Humphries, C., He, G., Hussein, S., Ke, L., Krogan, N., Li, Z., Levinson, J. N., Lu, H., Ménard, P., Munyana, C., Parsons, A. B., Ryan, O., Tonikian, R., Roberts, T., Sdicu, A.-M. M., Shapiro, J., Sheikh, B., Suter, B., Wong, S. L., Zhang, L. V., Zhu, H., Burd, C. G., Munro, S., Sander, C., Rine, J., Greenblatt, J., Peter, M., Bretscher, A., Bell, G., Roth, F. P., Brown, G. W., Andrews, B., Bussey, H., and Boone, C. (2004). Global mapping of the yeast genetic interaction network. *Science (New York, N. Y.)*, **303**(5659), 808–813.
- Walther, T. C. and Mann, M. (2010). Mass spectrometry-based proteomics in cell biology. *J Cell Biol*, **190**(4), 491–500.
- Wass, M. N., Fuentes, G., Pons, C., Pazos, F., and Valencia, A. (2011). Towards the prediction of protein interaction partners using physical docking. *Mol Syst Biol*, **7**, 469.
- Zamir, E. and Bastiaens, P. I. H. (2008). Reverse engineering intracellular biochemical networks. *Nat Chem Biol*, **4**(11), 643–7.

# Protein Hypernetworks: a Logic Framework for Interaction Dependencies and Perturbation Effects in Protein Networks (Supplementary Data)



**Figure S1:** Tableau for the propositional logic formula  $\phi = \neg A \wedge (A \vee B)$ . The path marked by  $\perp$  does not lead to a satisfying model  $\alpha$ , because it contains a contradiction between the assumptions  $\alpha \vdash \neg A$  and  $\alpha \vdash A$ . The path marked by  $\checkmark$  is free of contradictions, hence its generated model satisfies  $\phi$ .

This supplement contains background reference material on the Tableau Algorithm (Sec. S1), additional material on protein complex prediction with hypernetworks (Sec. S2), details on the generation of random constraints for null models (Sec. S3), supplementary material on the prediction of protein functional importance with the PIS defined in the main article (Sec. S4). We also provide software implementation details (Sec. S5), including resource consumption and details on the representation of constraints.

## S1 Background: Tableau Algorithm

A suitable method for finding satisfying models is the tableau calculus for propositional logic (Smullyan, 1995): For an input formula  $\phi$ , it generates a deductive tree (the *tableau*) of assumptions about  $\phi$ . Each assumption  $a$  in the tree can be made due to an assumption  $a'$  in an ancestral node. We say that  $a'$  *results in*  $a$ , and the generation of  $a$  out of  $a'$  is called *expansion* of  $a'$ . The propositional logic tableau algorithm generates satisfying models  $\alpha$  for  $\phi$ . We write  $\alpha \vdash \psi$  if  $\alpha$  satisfies a subformula  $\psi$ . The tableau algorithm now generates assumptions of the type  $\alpha \vdash \psi$  with  $\psi$  being a subformula of the input formula. That is, a conjunction  $\alpha \vdash \psi_1 \wedge \psi_2$  is expanded into  $\alpha \vdash \psi_1$  and  $\alpha \vdash \psi_2$  on the same path, and a disjunction  $\alpha \vdash \psi_1 \vee \psi_2$  results in branching into  $\alpha \vdash \psi_1$  and  $\alpha \vdash \psi_2$  (see Fig. S1).

Each path from the root to a leaf represents a model  $\alpha$ . If a path does not contain any contradictory assumptions, the model satisfies the input formula. Implementations of the tableau algorithm explore the tree in a depth-first way, and use backtracking once a contradiction occurs.

Different variations of the tableau algorithm exist. For example, one may be interested only in the decision “does a satisfying model exist?”, or the task could be to output an (arbitrary) satisfying model (if one exists), or to list all satisfying models. The latter is the task we face when enumerating minimal network states. In theory, the tableau algorithm exhibits an exponential worst case complexity, as it operates by complete enumeration of all cases with backtracking.

However, elaborate backtracking strategies can significantly reduce the running time in practice. Insights into such strategies and implementation details are provided by (Li, 2008). Also, faster heuristics exist, like GSAT (Selman *et al.*, 1992), but they are not adequate for the problem, as they do not guarantee a correct and complete answer.

For our purpose, the implementation has to ensure that

1. for each constraint  $q \Rightarrow \psi$  (i.e., disjunction  $\neg q \vee \psi$ ), the default case  $\neg q$  is explored first, and that  $\psi$  is expanded only if the constraint is necessarily active, in order to avoid artificially constrained models;
2. all satisfying models that comply with 1. are enumerated in the process.

In our application, the tableau algorithm can be expected to perform acceptably, since we solve only minimal network state formulas with a fixed structure (a conjunction of constraints) and expect most constraints to be of a simple form. In particular, we expect mostly mutual exclusive interactions, modeled by constraints of the form  $i \Rightarrow \neg j$ , and scaffold dependent interactions that can be represented by a constraint of the form  $i \Rightarrow j$ . To prove the performance of the tableau algorithm when all constraints are of this form, for a protein hypernetwork  $(P, I, C)$  we now show that it will need only  $\mathcal{O}(|C|)$  expansions to find a satisfying model. Since expansions generate the deductive tree, that also limits all tableau operations like backtracking or contradiction tests to be polynomial in  $|C|$ .

**Theorem S1.** *Let  $MNS_{(P,I,C)}(q)$  with  $q \in P \cup I$  be the minimal network state formula for a protein hypernetwork  $(P, I, C)$ . Assume that each constraint in  $C$  is of the form  $c = (q_1 \Rightarrow \ell)$  with a literal  $\ell \in \{q_2, \neg q_2\}$  and  $q_1, q_2 \in P \cup I$ . Then the tableau algorithm needs at most  $\mathcal{O}(|C|)$  expansions to find a satisfying model.*

*Proof.* We show that a constraint that is active cannot be rendered inactive again when assuming that the formula is satisfiable. Assume that an active constraint  $c = (q_1 \Rightarrow \ell)$  is the cause of a conflict, hence  $\ell$  contradicts some literal  $\ell'$ . Since we require above that the tableau explores the inactive case first, we know that  $\neg q_1$  caused a contradiction, too. We now assume that  $\ell$  is removed and we expand  $c$  to  $\neg q_1$  again to resolve the contradiction. Then, the formula is found to be not satisfiable, because  $\neg q_1$  can either contradict  $q$  or another constraint, in which case the argument can be applied recursively.

Now we show that each constraint is expanded at most two times. There are three cases: (1) The constraint is never activated; then only the inactive case is expanded and the constraint is expanded only once. (2) The constraint is activated immediately because  $\neg q_1$  leads to a conflict. This needs two expansions. (3) The constraint is first inactive and then activated because of a backtracking. This needs again two expansions. Hence, the tableau algorithm needs to perform  $1 + 2|C| = \mathcal{O}(|C|)$  expansions.  $\square$   $\square$

Note that  $MNS_{(P,I,C)}(q)$  with above simple constraints is essentially a Horn formula for which it is known that calculating a satisfying model has polynomial complexity with specialized algorithms (Dowling, 1984). However, proving the complexity of the general tableau algorithm for this case remains useful: While we expect most of our constraints to have this simple form, we cannot be sure for all of them. Hence it is reasonable to provide a computational approach that can handle full propositional logic, but will have comparable complexity to specialized horn formula algorithms in the majority of cases.

## S2 Complex Prediction in Hypernetworks

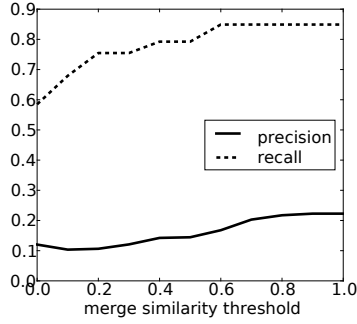
In this section, we provide more details on protein complex prediction, supplementing Section 3.3 of the main article.

\*Bioinformatics for High-Throughput Technologies, Algorithm Engineering, Computer Science 11, TU Dortmund, Germany

†Max Planck Institute of Molecular Physiology, Dortmund, Germany

‡to whom correspondence should be addressed





**Figure S2:** LCMA precision and recall for different merge similarity thresholds  $\omega$  on the CYGD protein network and complexes, as described in the main article.

### S2.1 Background: LCMA

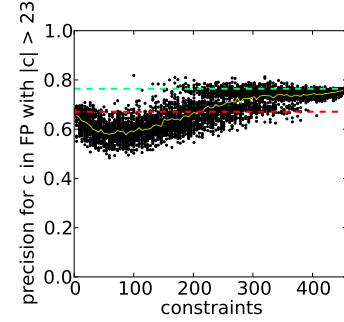
We briefly summarize the steps of the Local Clique Merging Algorithm (LCMA; Li *et al.* (2005)) as an exemplary complex prediction algorithm based on plain networks that can be improved by introducing protein hypernetworks. In a first step, LCMA finds a set of local cliques; second, it iteratively merges those with a significant overlap (given by a *merge similarity threshold*  $\omega$ ). The complex prediction consists of all merged cliques once no further merges happen or average density falls below 95% of the previous iteration. While the authors propose  $\omega = 0$ , we found LCMA to perform better with higher thresholds on the plain CYGD (Güldener *et al.*, 2005) network and complexes (Fig. S2). Now, a higher threshold  $\omega$  means that less clique merging is performed. The best choice of  $\omega = 1.0$  means that the clique merging step is not performed at all because only cliques that overlap by 100% (i.e. that are identical), would be merged. Therefore, we chose  $\omega = 0.4$  heuristically as a compromise between prediction quality and originally intended behaviour.

### S2.2 Effects of Gradually Introducing Constraints on Complex Prediction in Hypernetworks

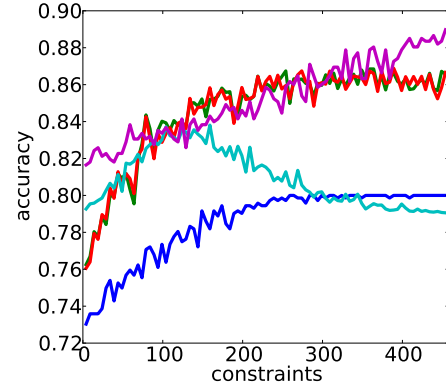
In the main article (Sec. 3.3), we showed that applying all 458 available constraints from Jung *et al.* (2010) resulted in an improved precision, while leaving the recall constant when predicting the CYGD complexes. Here we investigate the effect of a gradual application of constraints, in order to get an insight on their actual effects. Therefore, we randomly sampled subsets of all 458 available constraints of sizes between 4 (1% of 458) and 453 (99% of 458). More precisely, for each  $i \in \{1, 2, \dots, 99\}$ , we generated 50 independent samples of size  $i\%$  of 458 (rounded to the nearest integer).

**Precision and recall as a function of the number of applied constraints.** Fig. S3 shows the development of precision and recall as a function of the number of applied constraints. While the recall is independent of the number of applied constraints (Fig. S3a), high numbers of constraints ( $\geq 100$ ) consistently provided an improvement in the precision over the unconstrained instance with precision 0.15 (Fig. S3b). This indicates that constraining only about 1% of the interactions is already sufficient to robustly improve complex prediction. The maximum achieved precision then decreases gradually when applying more than 100 constraints and appears to reach a plateau upon using all available constraints. The minimum achieved precision rarely drops below the final precision value 0.20.

We offer the following explanation: Note that initially both the number of predicted complexes and the number of false positive predictions increase (Figs. S3c and S3d), but the latter one at a slower rate. Upon application of more interaction constraints both quantities reach a plateau and eventually decrease. How might this come about? A false positive complex may contain two interactions that are in reality mutually exclusive. The corresponding constraint might not be sampled when applying few constraints, resulting in one false positive prediction. When the number of constraints increases, the refinement step leads to two simultaneous protein sub-networks, on which again nearly the original complex without one of the exclusive interactions is predicted. Each of the two complexes may now be closer to a true benchmark complex, but the number



**Figure S4:** Complex prediction precision when removing all false positive complexes that contain at most 23 proteins. The red and green dashed lines mark the values obtained when none or all of the constraints are applied, respectively. The yellow line indicates the mean value for each number of applied constraints.



**Figure S5:** Matching accuracy for each CYGD complex over gradual application of constraints; mean values over 50 independent samples. Complexes with constant accuracy and complex Nop58/Nop56/Nop1 are not shown.

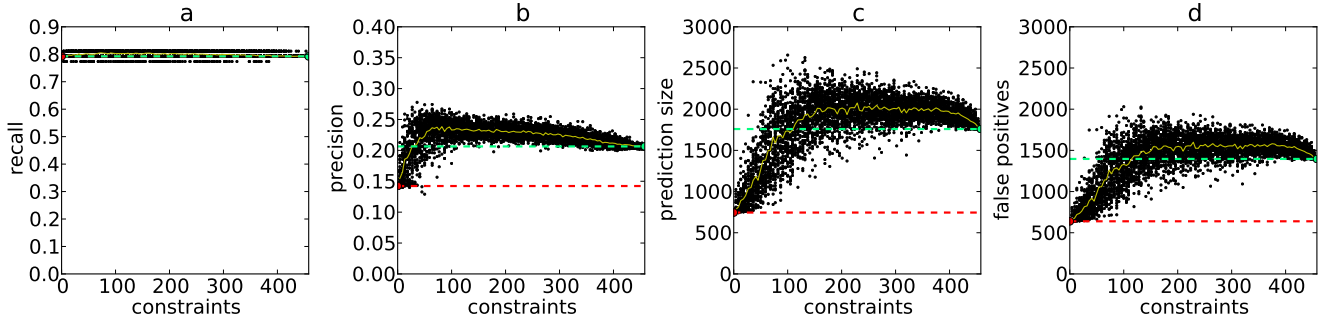
of constraints may still be too low to turn it into a true positive. Thus refinement of one false positive complex might initially lead to two or more false positive smaller complexes. This may underlie the observation that after an initial increase of the precision, showing a general beneficial effect of constraints, there is a stationary phase with even slightly decreasing precision.

Since the available constraints affect only less than 3% of all interactions, an important challenge is to extrapolate the development of the precision as a function of much higher numbers of constraints. It is rational to hypothesize that the precision will eventually start to increase upon constraining more interactions. Since we cannot prove this hypothesis directly at this point, we instead mimicked the effect of adding further constraints that may destroy small false positive complexes ( $< 24$  proteins) by artificially removing them from the prediction. After an initial decrease this leads to a precision increase when applying more than 100 constraints (Fig. S4). This observation is consistent with our hypothesis that the application of further constraints should lead to an increase of precision on all complexes.

**Accuracy on single complexes.** Complementary to the precision and recall of the whole prediction, we examined single complexes as well. As in Sec. 3.3 of the main article, we consider a predicted complex  $c$  to match a CYGD complex  $c'$  iff the matching accuracy  $\sqrt{(|c \cap c'|^2)/(|c| \cdot |c'|)}$  exceeds the threshold of  $\sqrt{0.2}$ .

When monitoring the accuracy of each CYGD complex  $c'$  while gradually introducing constraints, we would expect that the accuracy with its best matching prediction  $c$  remains constant or increases. Indeed, while the accuracy remains constant for most of the 55 complexes, it increases for four complexes, but there are also two complexes whose accuracy does not follow this expectation (Fig. S5; one of the latter ones not plotted).

The Nop58/Nop56/Nop1 complex (CYGD ID 440.12.30), one of the two complexes with decreasing accuracy, is not shown in Fig. S5



**Figure S3:** Complex prediction quality as a function of the number of applied constraints from 50 random samples for each step: (a) recall, (b) precision, (c) total number of predicted complexes, (d) false positive predictions. The red and green dashed lines mark the values obtained when none or all of the constraints are applied, respectively. The yellow line indicates the mean value for each number of applied constraints.

because it contains one constraint (Nop56 and Nop58 are competing on the same binding domain of Nop1) so that it disappears once this constraint is applied.

The Gim3/Gim5/Gim4/PAC10/YKE2 complex (CYGD ID 177, cyan in Figure S5) accuracy first increases then decreases, approximately returning to the initial value in the end. We identified the following constraints to hurt its accuracy:

$$\begin{aligned} \{SMC3, SMC3\} &\Rightarrow \neg \{SMC1, SMC3\} \\ \{ARP6, SWD3\} &\Rightarrow \neg \{CLA4, SWD3\} \\ \{SKP1, CDC53\} &\Rightarrow \neg \{MET30, CDC53\}. \end{aligned}$$

These findings do not necessarily imply that those constraints are wrong. Rather they are hindering the heuristic LCMA in predicting the two complexes by altering the density of the corresponding regions in the simultaneous protein subnetwork. This shows that predicting complexes by density – while it seems a good strategy in general – does indeed fail for single cases.

### S3 Generation of Random Constraints

It is important to compare the effects of (presumably) true known constraints with the effect of random constraints in order to show that observed effects are not simply due to applying constraints *per se*. Here, we specify how we generate random constraints of the type “mutually exclusive interaction”.

To generate a random constraint, we randomly choose a protein  $p_1 \in P$  network, and randomly select two different neighbours  $p_2, p_3 \in P$ . We interpret  $p_1$  as the host protein and  $p_2, p_3$  as two proteins competing on the same binding domain of  $p_1$ . Thereby we obtain the constraints  $\{p_1, p_2\} \Rightarrow \neg \{p_1, p_3\}$  and  $\{p_1, p_3\} \Rightarrow \neg \{p_1, p_2\}$ .

For the CYGD hypernetwork (Sec. 3.3 of the paper), we iteratively generate 458 of these constraint pairs (we refer to such a pair simply as one constraint). By independently repeating this process  $n$  times, we gain  $n$  independent samples of 458 random constraints.

### S4 Protein Functional Importance Prediction with Hypernetworks

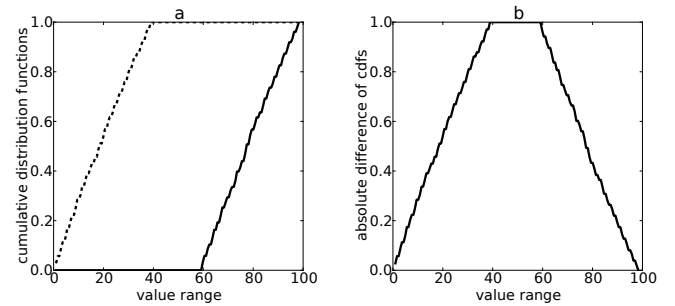
In this section, we provide more details on protein complex prediction, supplementing Section 4 of the main article.

#### S4.1 Phenotypes of Null Mutants in the Saccharomyces Genome Database

We used the Saccharomyces Genome Database (SGD) (Cherry *et al.*, 1998) to classify perturbations as lethal/sick. SGD collects the generated phenotypes of perturbation experiments for most proteins that are also in the CYGD, our selected benchmark. SGD phenotypes are provided in a standardized way, a complete list is provided by the *Ontology Lookup Service* (Cote *et al.*, 2006). To fit our modelling of perturbation and the notion of functional importance in the main article, we considered only “null mutant” perturbations rather than e.g. overexpression experiments. Table S1 shows those phenotypes that were counted as lethal/sick. In contrast, the class of viable perturbations contains all that are annotated with the

**Table S1:** SGD phenotypes selected to be classified as lethal/sick. A phenotype is composed by an observable and a qualifier.

observable	qualifier
cell death	increased, increased rate
apoptosis	increased, increased rate
autolysis	increased, increased rate
cell lysis	increased, increased rate
necrotic cell death	increased, increased rate
competetive fitness	decreased
viability	decreased
vegetative growth	decreased
inviable	-



**Figure S6:** Example for cdf based separation analysis of two completely separated distributions: uniform distribution  $U_1$  on  $[0, 40]$  and uniform distribution  $U_2$  on  $[60, 100]$ . (a) cumulative distribution functions (cdfs) for  $U_1$  (dashed) and  $U_2$  (solid). x-axis: value range of the distributions. E.g. at  $x = 50$ , the cdf of  $U_1$  has reached 1, while that of  $U_2$  is still zero. (b) absolute difference of the two cdfs.

phenotype “viable” and are not contained in the class of lethal/sick ones.

#### S4.2 Analysis of cumulative PIS distributions

To find out about the capability of PIS to indicate functional important proteins, we analysed its distribution for disjoint classes like lethal/sick and viable. Therefore, we calculated empirical cumulative distribution functions (cdfs; normed cumulative histograms) for both classes. Maximum separation is provided if the “lethal/sick” cdf does not increase over 0 before the “viable” histogram reaches 1 (Fig. S6a). No separation means that both cdfs have the same values for each score. To better compare several cdf pairs, we calculate the absolute difference between the two cdfs (Fig. S6b). The higher the absolute difference, the better is the separation. If the absolute difference reaches 1.0 at any point, the distributions are completely separated.

**Table S2:** Representation of possible interaction constraints in structured text. Keywords and proteins are annotated with a defined ontology term id. This way, finding and parsing constraints is improved while human readability is maintained.

constraint	structured text
mutually exclusive interactions	x (uniprotkb:x) competes (MI:0941) with y (uniprotkb:y) for interaction (MI:0407) with z (uniprotkb:z).
negative allosteric regulation by protein binding	interaction (MI:0407) between x (uniprotkb:x) and y (uniprotkb:y) allosterically (SBO:0000239) inhibits (SBO:0000407) the interaction (MI:0407) of y (uniprotkb:y) with z (uniprotkb:z).
positive allosteric regulation by protein binding	interaction (MI:0407) between x (uniprotkb:x) and y (uniprotkb:y) allosterically (SBO:0000239) activates (SBO:0000461) the interaction (MI:0407) of y (uniprotkb:y) with z (uniprotkb: z).
negative regulation by phosphorylation	interaction (MI:0407) between x (uniprotkb:x) and y (uniprotkb:y) is inhibited (SBO:0000407) if x (uniprotkb:x) is phosphorylated (GO:0016310) on residue <i>i</i> .
positive regulation by phosphorylation	interaction (MI:0407) between x (uniprotkb:x) and y (uniprotkb:y) is activated (SBO:0000461) if x (uniprotkb:x) is phosphorylated (GO:0016310) on residue <i>i</i> .

## S5 Software Implementation

We implemented protein hypernetworks as a JAVA<sup>TM</sup> based software suite. The suite consists of *ProteinHypernetworkEditor* that allows the definition and editing of protein hypernetworks, and *ProteinHypernetwork* that implements the prediction methods presented in the main article. Further, both tools provide a graphical user interface and extensive visualization and import/export capabilities. The software suite can be obtained at <http://www.rahmannlab.de/research/hypernetworks>.

### S5.1 Representation of Constraints

An important challenge is the definition of a widely accepted format for the interchange of interaction dependencies or constraints. While SBML (Hucka *et al.*, 2003) is suitable in principle, it provides a biochemical view of interactions and therefore contains overhead that is unnecessary for the definition of a protein hypernetwork. Instead we propose a two level approach for the interchange of constraints.

**Level 1: Structured Text.** Ceol *et al.* (2008b) proposed a machine readable structured abstract that should be published along with papers on protein interactions to allow automated curation (see also (Leitner *et al.*, 2010)). The format combines human-readable sentences with machine readable annotation, and is already capable of representing the expected types of constraints. For example, a pair of mutual exclusive interactions (as reported by Jung *et al.* (2010)) can be represented as follows: "ARC40 (uniprotkb:P38328) is a *competitor* (MI:0941) of BEM2 (uniprotkb:P39960) for *interaction* (MI:0317) with CLA4 (uniprotkb:P48562)". Table S2 provides a generalized representation for the major types of interaction constraints.

**Level 2: HypernetworkML.** With the hypernetwork markup language (HypernetworkML, Köster (2011)) we provide an XML-based file format (Bray *et al.*, 1998) that is more suitable for possible large-scale studies providing many constraints at once and for permanent storage of the data. HypernetworkML is a combination of two established XML based formats: Interactions and proteins are represented as a graph using GraphML (Brandes *et al.*, 2002) whereas embedded MathML (Carlisle *et al.*, 2003) is used for a propositional logic definition of constraints. Hence, HypernetworkML is capable to represent a complete protein hypernetwork while maintaining compatibility with known standards.

### S5.2 Resource Consumption

A complex prediction on the defined yeast hypernetwork takes 12 seconds using all 4 cores of an Intel<sup>®</sup> Core<sup>TM</sup> i5 CPU with 2.8GHz. The prediction of the PIS of 4579 single protein perturbations takes 8 seconds. In both cases the software uses approximately 750MB RAM during prediction.